```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

FILEID**RM3OPEN

```
RRRRRRRR    MM      MM    333333      000000    PPPPPPPP    EEEEEEEEEE  NN      NN
RRRRRRRR    MM      MM    333333      000000    PPPPPPPP    EEEEEEEEEE  NN      NN
RR      RR  MMMM  MMMM  33      33  00      00  PP      PP  EE          NN      NN
RR      RR  MMMM  MMMM  33      33  00      00  PP      PP  EE          NN      NN
RR      RR  MM  MM  MM        33    00      00  PP      PP  EE          NNNN    NN
RR      RR  MM  MM  MM        33    00      00  PP      PP  EE          NNNN    NN
RRRRRRRR    MM      MM      33      00      00  PPPPPPPP    EEEEEEEE    NN  NN  NN
RRRRRRRR    MM      MM      33      00      00  PPPPPPPP    EEEEEEEE    NN  NN  NN
RR  RR      MM      MM        33    00      00  PP          EE          NN    NNNN
RR  RR      MM      MM        33    00      00  PP          EE          NN    NNNN
RR    RR    MM      MM  33    33    00      00  PP          EE          NN      NN      ....
RR    RR    MM      MM  33    33    00      00  PP          EE          NN      NN      ....
RR      RR  MM      MM    333333      000000    PP          EEEEEEEEEE  NN      NN      ....
RR      RR  MM      MM    333333      000000    PP          EEEEEEEEEE  NN      NN      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

RM3
V04

```
;     1          0001  0 MODULE RM3OPEN (LANGUAGE (BLISS32) ,
;     2          0002  0                  IDENT = 'V04-000'
;     3          0003  0                  ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                  *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
```

```
;    29     0028  1 !++
;    30     0029  1 !
;    31     0030  1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;    32     0031  1 !
;    33     0032  1 ! ABSTRACT:
;    34     0033  1 !                organization independent code for indexed file open
;    35     0034  1 !
;    36     0035  1 !
;    37     0036  1 ! ENVIRONMENT:
;    38     0037  1 !
;    39     0038  1 !                VAX/VMS OPERATING SYSTEM
;    40     0039  1 !
;    41     0040  1 !--
;    42     0041  1
;    43     0042  1 !
;    44     0043  1 ! AUTHOR:       Wendy Koenig     CREATION DATE:         24-MAR-78  13:20
;    45     0044  1 !
;    46     0045  1 !
;    47     0046  1 ! MODIFIED BY:
;    48     0047  1 !
;    49     0048  1 !       V03-016 RAS0284         Ron Schaefer            30-Mar-1984
;    50     0049  1 !               Fix STV value on error pachs for RMS$_RPL and RMS$_WPL errors.
;    51     0050  1 !
;    52     0051  1 !       V03-015 DAS0001         David Solomon           25-Mar-1984
;    53     0052  1 !               Fix broken branch to RM$ALDBUF.
;    54     0053  1 !
;    55     0054  1 !       V03-014 SHZ0001         Stephen H. Zalewski     27-Feb-1984
;    56     0055  1 !               If you allocate a BDB, you MUST bump the local buffer count
;    57     0056  1 !               (IFB$W_AVLCL).
;    58     0057  1 !
;    59     0058  1 !       V03-013 JWT0141         Jim Teague              11-Nov-1983
;    60     0059  1 !               Oops, IFB$V_RUM changed to IFB$V_ONLY_RU
;    61     0060  1 !
;    62     0061  1 !       V03-012 JWT0140         Jim Teague              11-Nov-1983
;    63     0062  1 !               Must check more than one RU bit, as was done in
;    64     0063  1 !               V03-010.
;    65     0064  1 !
;    66     0065  1 !       V03-011 MCN0013         Maria del C. Nasr       24-Feb-1983
;    67     0066  1 !               Reorganize linkages.
;    68     0067  1 !
;    69     0068  1 !       V03-010 TMK0005         Todd M. Katz            20-Jan-1983
;    70     0069  1 !               Add support for RMS Journalling and Recovery of ISAM files.
;    71     0070  1 !               For $OPEN this boils down to not allowing a prologue 1 or 2
;    72     0071  1 !               file to be opened if it is marked for any type of journalling.
;    73     0072  1 !
;    74     0073  1 !       V03-009 KBT0464         Keith B. Thompson       13-Jan-1983
;    75     0074  1 !               Get BKS from key descriptors to aviod LCL bugchecks due
;    76     0075  1 !               to wrong file header info
;    77     0076  1 !
;    78     0077  1 !       V03-008 KBT0460         Keith B. Thompson       12-Jan-1983
;    79     0078  1 !               Allocate a buffer for reading in prologue (it use to use
;    80     0079  1 !               the buffer allocated for the fwa)
;    81     0080  1 !
;    82     0081  1 !       V03-007 KBT0225         Keith B. Thompson       23-Aug-1982
;    83     0082  1 !               Reorganize psects
;    84     0083  1 !
;    85     0084  1 !       V03-006 TMK0004         Todd M. Katz            18-Aug-1982
```

```
;    86          0085  1 !
;    87          0086  1 !       V03-006 TMK0004         Todd M. Katz            18-Aug-1982
;    88          0087  1 !               Allow prologue 3 files with alternate indicies to be opened.
;    89          0088  1 !
;    90          0089  1 !       V03-005 TMK0003         Todd M. Katz            01-Jul-1982
;    91          0090  1 !               Implement RMS cluster solution for next record positioning.
;    92          0091  1 !               This emans that RMS no longer has to zero the pointer to the
;    93          0092  1 !               NRP cell in the IFAB, IFB$L_NRP_PTR, because the next record
;    94          0093  1 !               positioning context is now kept locally in the IRAB instead
;    95          0094  1 !               of in a separate systemwide location.
;    96          0095  1 !
;    97          0096  1 !       V03-004 MCN0012         Maria del C. Nasr       29-Jun-1982
;    98          0097  1 !               Allow different key data types for prologue 3 files.
;    99          0098  1 !               This undoes part of TMK0002.
;   100          0099  1 !
;   101          0100  1 !       V03-003 KBT0054         Keith B. Thompson       8-Jun-1982
;   102          0101  1 !               Allocate index blocks on all but BIO or UFO opens
;   103          0102  1 !
;   104          0103  1 !       V03-002 TMK0002         Todd M. Katz            06-May-1982
;   105          0104  1 !               I added code to prevent prologue 3 files with key types
;   106          0105  1 !               other than string and/or alternate indicies from being opened.
;   107          0106  1 !               This code is required for V3A - V3B compatibility, it will go
;   108          0107  1 !               out as a V3.1 patch, and it must be removed for V3B when
;   109          0108  1 !               alternate data types and indicies are supported. The error that
;   110          0109  1 !               will be returned is: error in prologue version.
;   111          0110  1 !
;   112          0111  1 !               I also fixed up some of the error paths which were not
;   113          0112  1 !               releasing all accessed VBNs of the file before returning
;   114          0113  1 !               their appropriate error.
;   115          0114  1 !
;   116          0115  1 !       V03-001 TMK0001         Todd M. Katz            24-Mar-1982
;   117          0116  1 !               Change all references from IFB$B_KBUFSZ to IFB$W_KBUFSZ.
;   118          0117  1 !
;   119          0118  1 !       V02-020 CDS0005         C D Saether             5-Feb-1982
;   120          0119  1 !               Back out V02-019.  GBC is now a record attribute.
;   121          0120  1 !
;   122          0121  1 !       V02-019 CDS0004         C D Saether             3-Jan-1982
;   123          0122  1 !               Return GBC field from prologue.
;   124          0123  1 !
;   125          0124  1 !       V02-018 CDS0003         C D Saether             9-Aug-1981
;   126          0125  1 !               Use alternate linkage declaration for RELEASE.
;   127          0126  1 !
;   128          0127  1 !       V02-017 CDS0002         C D Saether             16-Jul-1981
;   129          0128  1 !               Remove check for ppf file.
;   130          0129  1 !
;   131          0130  1 !       V02-016 MCN0011         Maria del C. Nasr       05-Jun-1981
;   132          0131  1 !               Make keybuffer size 2 bytes longer for compressed indexes,
;   133          0132  1 !               and primary key.
;   134          0133  1 !
;   135          0134  1 !       V02-015 PSK0002         P S Knibbe              20-Apr-1981
;   136          0135  1 !               Change some variable names
;   137          0136  1 !
;   138          0137  1 !       V02-014 PSK0001         P S Knibbe              17-Mar-1981
;   139          0138  1 !               Change the prologue number check to allow prologue 3
;   140          0139  1 !               Change check_two to make sure that at least two index records
;   141          0140  1 !               can fit into an index bucket.
;   142          0141  1 !
```

```
;  143          0142  1 !       V02-013 REFORMAT        R A SCHAEFER            23-Jul-1980     14:09
;  144          0143  1 !               Reformat the source
;  145          0144  1 !
;  146          0145  1 !       V02-012 CDS0001         C D SAETHER             13-MAR-1980
;  147          0146  1 !               fix V011 fix to check bio in ifab, not fab
;  148          0147  1 !
;  149          0148  1 !       V02-011 RAS0000         Ron Schaefer            27-NOV-79       09:30
;  150          0149  1 !               Allow BIO access to any device (i.e. magtape), do not read
;  151          0150  1 !               prolog if so.
;  152          0151  1 !
;  153          0152  1 !       V02-010 CDS0000         Chris Saether,          26-jun-79  17:55
;  154          0153  1 !               don't allocate stuff if UFO set
;  155          0154  1 !
;  156          0155  1 !****
;  157          0156  1
;  158          0157  1 LIBRARY 'RMSLIB:RMS';
;  159          0158  1
;  160          0159  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;  161          0224  1
;  162          0225  1 ! define default psects for code
;  163          0226  1 !
;  164          0227  1 PSECT
;  165          0228  1     CODE = RM$RMS3(PSECT_ATTR),
;  166          0229  1     PLIT = RM$RMS3(PSECT_ATTR);
;  167          0230  1
;  168          0231  1 ! define linkages
;  169          0232  1 !
;  170          0233  1 LINKAGE
;  171          0234  1     L_ALDBUF,
;  172          0235  1     L_CACHE,
;  173          0236  1     L_CHKSUM,
;  174          0237  1     L_FABREG,
;  175          0238  1     L_LINK_7_10_11,
;  176          0239  1     L_RELEASE_FAB,
;  177          0240  1     RL$CHECK_TWO        = JSB (REGISTER = 6) :
;  178          0241  1                           GLOBAL (R_FAB,R_IFAB);
;  179          0242  1
;  180          0243  1 ! forward routine
;  181          0244  1 !
;  182          0245  1
;  183          0246  1 FORWARD ROUTINE
;  184          0247  1     RM$OPEN3B           : RL$FABREG,
;  185          0248  1     CHECK_TWO           : RL$CHECK_TWO;
;  186          0249  1
;  187          0250  1 ! external routines
;  188          0251  1 !
;  189          0252  1 EXTERNAL ROUTINE
;  190          0253  1     RM$ALDBUF           : RL$ALDBUF ADDRESSING_MODE( LONG_RELATIVE ),
;  191          0254  1     RM$CHKSUM           : RL$CHKSUM,
;  192          0255  1     RM$CACHE            : RL$CACHE,
;  193          0256  1     RM$CLOSE3           : RL$LINK_7_10_11,
;  194          0257  1     RM$RELEASE          : RL$RELEASE_FAB,
;  195          0258  1     RM$AL_KEY_DESC      : RL$LINK_7_TO_11;
;  196          0259  1
```

```
;  198     0260  1 %SBTTL  'RM$OPEN3B'
;  199     0261  1 GLOBAL ROUTINE RM$OPEN3B : RL$FABREG =
;  200     0262  1 !++
;  201     0263  1 !
;  202     0264  1 ! FUNCTIONAL DESCRIPTION:
;  203     0265  1 !
;  204     0266  1 !               This routine performs the file open functions that are
;  205     0267  1 !               specific to the indexed file organization, including:
;  206     0268  1 !
;  207     0269  1 !               1 -- reading in the prologue
;  208     0270  1 !                       and setting up various fields in the FAB and IFAB
;  209     0271  1 !               2 -- setting up the index descriptors
;  210     0272  1 !                       (linked off the IFAB) and counting the keys
;  211     0273  1 !               3 -- determining the size of the  key buffers
;  212     0274  1 !                       and setting kbufsz (IFAB) appropriately
;  213     0275  1 !
;  214     0276  1 !
;  215     0277  1 ! CALLING SEQUENCE:
;  216     0278  1 !
;  217     0279  1 !       enters via case branch from RMS$OPEN and jsb from RM$OPEN3
;  218     0280  1 !       returns via rsb to RM$COPRTN.
;  219     0281  1 !
;  220     0282  1 ! INPUT PARAMETERS:
;  221     0283  1 !       none
;  222     0284  1 !
;  223     0285  1 ! IMPLICIT INPUTS:
;  224     0286  1 !
;  225     0287  1 !       R11     IMPURE AREA address
;  226     0288  1 !       R9      IFAB address
;  227     0289  1 !       R8      FAB address
;  228     0290  1 !       the contents of the FAB
;  229     0291  1 !
;  230     0292  1 ! OUTPUT PARAMETERS:
;  231     0293  1 !       none
;  232     0294  1 !
;  233     0295  1 ! IMPLICIT OUTPUTS:
;  234     0296  1 !
;  235     0297  1 !       R10 is the address of the IFAB
;  236     0298  1 !       various fields in the IFAB and FAB are initialized
;  237     0299  1 !       index descriptors are allocated
;  238     0300  1 !
;  239     0301  1 ! ROUTINE VALUE:
;  240     0302  1 !
;  241     0303  1 !       standard rms, in particular SUC,PLG,RPL,IFA,KSI,ENV
;  242     0304  1 !
;  243     0305  1 ! SIDE EFFECTS:
;  244     0306  1 !
;  245     0307  1 !       May wait quite some time for prologue to become free initially.
;  246     0308  1 !       Allocates index descriptors
;  247     0309  1 !       In the case of an error, key descriptors are deallocated
;  248     0310  1 !       R1 - R5  may be destroyed
;  249     0311  1 !
;  250     0312  1 !--
;  251     0313  1
;  252     0314  2     BEGIN
;  253     0315  2
;  254     0316  2     ! Define common registers
```

```
;  255          0317  2     !
;  256          0318  2     EXTERNAL REGISTER
;  257          0319  2         COMMON_FAB_STR;
;  258          0320  2
;  259          0321  2     GLOBAL REGISTER
;  260          0322  2         COMMON_IO_STR;
;  261          0323  2
;  262          0324  2     IFAB = .IFAB_FILE;
;  263          0325  2
;  264          0326  2     ! Have to zero this since it has a conflicting earlier use in the parse
;  265          0327  2     !
;  266          0328  2     IFAB [ IFB$L_IDX_PTR ] = 0;
;  267          0329  2
;  268          0330  2     ! Allocate a BDB in preparation for reading in the prologue.  Even if we
;  269          0331  2     ! do not use it here, it may be used for XAB processing later on.
;  270          0332  2     !
;  271          0333  2     RETURN_ON_ERROR( RM$ALDBUF( 512 ) );                ! Get a BDB.
;  272          0334  2     IFAB[IFB$W_AVLCL] = .IFAB[IFB$W_AVLCL] + 1;         ! Bump the local buffer count.
;  273          0335  2
;  274          0336  2     ! if UFO or BIO open then quit right here before descriptors get allocated
;  275          0337  2     !
;  276          0338  2     IF  .FAB [ FAB$V_UFO ] OR .IFAB [ IFB$V_BIO ]
;  277          0339  2     THEN
;  278          0340  2         RETURN RMSSUC( SUC );
;  279          0341  2
;  280          0342  2     ! Read in the prologue 1 block which also has the first key descriptor
;  281          0343  2     !
;  282        P 0344  2     RETURN_ON_ERROR( CACHE( 1,512 ),
;  283        P 0345  2                      BEGIN
;  284        P 0346  2                      IF .FAB [FAB$L_STV] EQL 0
;  285        P 0347  2                      THEN
;  286        P 0348  2                          FAB [FAB$L_STV] = .STATUS OR 1^16;
;  287        P 0349  2                      STATUS = RMSERR (RPL)
;  288          0350  2                      END );
;  289          0351  2
;  290          0352  2     RETURN_ON_ERROR( RM$CHKSUM() );
;  291          0353  2
;  292          0354  2     ! Check for correct prologue version
;  293          0355  2     !
;  294          0356  2     IF .BKT_ADDR [ PLG$W_VER_NO ] GTRU PLG$C_VER_3
;  295          0357  2     THEN
;  296          0358  3         BEGIN
;  297          0359  3         RM$RELEASE(0);
;  298          0360  4         RETURN RMSERR( PLV )
;  299          0361  2         END;
;  300          0362  2
;  301          0363  2     ! Do not allow this file to be opened if it is a prologue 1 or 2 file, and
;  302          0364  2     ! any type of RMS Journalling is enabled.
;  303          0365  2     !
;  304          0366  2     IF  .BKT_ADDR[PLG$W_VER_NO] LSSU PLG$C_VER_3
;  305          0367  2          AND
;  306          0368  3         (.IFAB[IFB$V_RU]
;  307          0369  3                  OR
;  308          0370  3                 .IFAB[IFB$V_ONLY_RU]
;  309          0371  3                  OR
;  310          0372  3                 .IFAB[IFB$V_AT]
;  311          0373  3                  OR
```

```
;   312     0374  3                   .IFAB[IFB$V_BI]
;   313     0375                        OR
;   314     0376                        .IFAB[IFB$V_AI])
;   315     0377  2             THEN
;   316     0378  2                 BEGIN
;   317     0379  2                 RM$RELEASE(0);
;   318     0380  2                 RETURN RMSERR(ENV);
;   319     0381  2                 END;
;   320     0382  2
;   321     0383  2             ! We now have a good prologue in memory
;   322     0384  2             !
;   323     0385  2             IFAB [ IFB$B_PLG_VER ] = .BKT_ADDR [ PLG$W_VER_NO ];
;   324     0386  2             IFAB [ IFB$B_AVBN ] = .BKT_ADDR [ PLG$B_AVBN ];
;   325     0387  2             IFAB [ IFB$B_AMAX ] = .BKT_ADDR [ PLG$B_AMAX ];
;   326     0388  2             IFAB [ IFB$W_FFB ] = 0;
;   327     0389  2
;   328     0390  2             ! Allocate and count index descriptors, determine size of key buffers
;   329     0391  2             !
;   330     0392  2             BEGIN
;   331     0393  2
;   332     0394  2             GLOBAL REGISTER
;   333     0395  2                 R_IDX_DFN;
;   334     0396  2
;   335     0397  2             LOCAL
;   336     0398  2                 IDX_COMPR,
;   337     0399  2                 KEY_DESC        : REF BBLOCK;
;   338     0400  2
;   339     0401  2             ! Index descriptor for primary key the primary key obviously is the largest
;   340     0402  2             ! to date, so set kbufsz
;   341     0403  2             !
;   342     0404  2             IFAB [ IFB$W_KBUFSZ ] = .BKT_ADDR [ KEY$B_KEYSZ ];
;   343     0405  2
;   344     0406  2             ! Start off finding the largest bucket size for key 0
;   345     0407  2             !
;   346     0408  2             IF .BKT_ADDR [ KEY$B_IDXBKTSZ ] GTRU .BKT_ADDR [ KEY$B_DATBKTSZ ]
;   347     0409  2             THEN
;   348     0410  2                 IFAB [ IFB$B_BKS ] = .BKT_ADDR [ KEY$B_IDXBKTSZ ]
;   349     0411  2             ELSE
;   350     0412  2                 IFAB [ IFB$B_BKS ] = .BKT_ADDR [ KEY$B_DATBKTSZ ];
;   351     0413  2
;   352     0414  2             ! Assume no compression
;   353     0415  2             !
;   354     0416  2             IDX_COMPR = 0;
;   355     0417  2
;   356     0418  2             ! Allocate the primary key descriptor
;   357     0419  2             !
;   358     0420  2             RETURN_ON_ERROR( RM$AL_KEY_DESC( .BKT_ADDR,1,0 ), RM$RELEASE(0) );
;   359     0421  2
;   360     0422  2             IFAB [ IFB$B_NUM_KEYS ] = 1;
;   361     0423  2
;   362     0424  2             KEY_DESC = .BKT_ADDR;
;   363     0425  2
;   364   P 0426  2             RETURN_ON_ERROR( CHECK_TWO(),
;   365   P 0427  2                              BEGIN
;   366   P 0428  2                              RM$CLOSE3();
;   367   P 0429  2                              RM$RELEASE(0)
;   368     0430  2                              END );
```

```
;  369          0431  3
;  370          0432  3      ! If the index or primary key is compressed, set flag.
;  371          0433  3      !
;  372          0434  3      IF .KEY_DESC [ KEY$V_IDX_COMPR ] OR .KEY_DESC [ KEY$V_KEY_COMPR ]
;  373          0435  3      THEN
;  374          0436  3          IDX_COMPR = 1;
;  375          0437  3
;  376          0438  3      ! Get index descriptors for all other keys, block by block
;  377          0439  3      !
;  378          0440  3      WHILE .KEY_DESC [ KEY$L_IDXFL ] NEQ 0
;  379          0441  3      DO
;  380          0442  4          BEGIN
;  381          0443  4
;  382          0444  4          LOCAL
;  383          0445  4              VBN,
;  384          0446  4              OFFSET;
;  385          0447  4
;  386          0448  4          ! Save the vbn and the offset which is in this block
;  387          0449  4          !
;  388          0450  4          VBN = .KEY_DESC [ KEY$L_IDXFL ];
;  389          0451  4          OFFSET = .KEY_DESC [ KEY$W_NOFF ];
;  390          0452  4
;  391          0453  4          ! Release current block and the new one
;  392          0454  4          !
;  393          0455  4          RETURN_ON_ERROR( RM$RELEASE(0) );
;  394          0456  4
;  395        P 0457  4          RETURN_ON_ERROR( CACHE( .VBN,512 ),
;  396        P 0458  4                           BEGIN
;  397        P 0459  4                           IF .FAB [FAB$L_STV] EQL 0
;  398        P 0460  4                           THEN
;  399        P 0461  4                               FAB [FAB$L_STV] = .STATUS OR 1^16;
;  400        P 0462  4                           STATUS = RMSERR (RPL)
;  401          0463  4                           END );
;  402          0464  4
;  403          0465  4
;  404          0466  4          RETURN_ON_ERROR( RM$CHKSUM() );
;  405          0467  4
;  406          0468  4          ! Loop for all of the key descriptors in this block
;  407          0469  4          !
;  408          0470  4          DO
;  409          0471  5              BEGIN
;  410          0472  5
;  411          0473  5              ! Set the pointer to the new key descriptor
;  412          0474  5              !
;  413          0475  5              KEY_DESC = .BKT_ADDR + .OFFSET;
;  414          0476  5
;  415        P 0477  5              RETURN_ON_ERROR( CHECK_TWO(),
;  416        P 0478  5                               BEGIN
;  417        P 0479  5                               RM$CLOSE3();
;  418        P 0480  5                               RM$RELEASE(0)
;  419          0481  5                               END );
;  420          0482  5
;  421          0483  5              ! We have a good one so count it
;  422          0484  5              !
;  423          0485  5              IFAB [ IFB$B_NUM_KEYS ] = .IFAB [ IFB$B_NUM_KEYS ] + 1;
;  424          0486  5
;  425          0487  5              ! Set the largest key size
```

```
;  426     0488  5          !
;  427     0489  5          IF .KEY_DESC [ KEY$B_KEYSZ ] GTRU .IFAB [ IFB$W_KBUFSZ ]
;  428     0490  5          THEN
;  429     0491  5              IFAB [ IFB$W_KBUFSZ ] = .KEY_DESC [ KEY$B_KEYSZ ];
;  430     0492  5
;  431     0493  5          ! Set the largest bucket size
;  432     0494  5          !
;  433     0495  5          IF .KEY_DESC [ KEY$B_IDXBKTSZ ] GTRU .IFAB [ IFB$B_BKS ]
;  434     0496  5          THEN
;  435     0497  5              IFAB [ IFB$B_BKS ] = .KEY_DESC [ KEY$B_IDXBKTSZ ];
;  436     0498  5
;  437     0499  5          IF .KEY_DESC [ KEY$B_DATBKTSZ ] GTRU .IFAB [ IFB$B_BKS ]
;  438     0500  5          THEN
;  439     0501  5              IFAB [ IFB$B_BKS ] = .KEY_DESC [ KEY$B_DATBKTSZ ];
;  440     0502  5
;  441     0503  5          ! This index descriptor is ok so allocate one in memory
;  442     0504  5          !
;  443   P 0505  5          RETURN_ON_ERROR( RM$AL_KEY_DESC( .KEY_DESC,.VBN,.OFFSET ),
;  444     0506  5                           RM$RELEASE(0) );
;  445     0507  5
;  446     0508  5          ! If there is compression on note it
;  447     0509  5          !
;  448     0510  5          IF .KEY_DESC [ KEY$V_IDX_COMPR ]
;  449     0511  5          THEN
;  450     0512  5              IDX_COMPR = 1;
;  451     0513  5
;  452     0514  5          ! Get the offset to the next key descriptor
;  453     0515  5          !
;  454     0516  5          OFFSET = .KEY_DESC [ KEY$W_NOFF ]
;  455     0517  5
;  456     0518  5          END
;  457     0519  5
;  458     0520  5              ! Leave the loop if the next key descriptor is in another block
;  459     0521  5              !
;  460     0522  4              UNTIL .KEY_DESC [ KEY$L_IDXFL ] NEQ .VBN
;  461     0523  4
;  462     0524  3              END;
;  463     0525  3
;  464     0526  3          ! If any of the keys have the index compressed, then increase the buffer
;  465     0527  3          ! size by two bytes, to store the length and compression counts.
;  466     0528  3          !
;  467     0529  3          IF .IDX_COMPR
;  468     0530  3          THEN
;  469     0531  3              IFAB [ IFB$W_KBUFSZ ] = .IFAB [ IFB$W_KBUFSZ ] + 2
;  470     0532  3
;  471     0533  2          END;
;  472     0534  2
;  473     0535  2          RETURN_ON_ERROR( RM$RELEASE(0) );
;  474     0536  2
;  475     0537  3          RETURN RMSSUC()
;  476     0538  3
;  477     0539  1          END;


                                                    .TITLE  RM3OPEN
                                                    .IDENT  \V04-000\
```

```
                                                      .EXTRN  RMSALDBUF, RMSCHKSUM
                                                      .EXTRN  RMSCACHE, RMSCLOSE3
                                                      .EXTRN  RMSRELEASE, RMSAL_KEY_DESC

                                                      .PSECT  RMSRMS3,NOWRT,  GBL,  PIC,2

                         00FC   BF  BB 00000 RMSOPEN3B::
                                                      PUSHR   #^M<R2,R3,R4,R5,R6,R7>                    ; 0261
                 5E             14  C2 00004          SUBL2   #20, SP
                 5A             59  D0 00007          MOVL    IFAB_FILE, IFAB                           ; 0324
                         00AC   CA  D4 0000A          CLRL    172(IFAB)                                 ; 0328
                 55      0200   8F  3C 0000E          MOVZWL  #512, R5                                  ; 0333
                    00000000G   EF  16 00013          JSB     RMSALDBUF                                 ;
                 68             50  E9 00019          BLBC    STATUS, 7$                                ;
                         0084   CA  B6 0001C          INCW    132(IFAB)                                 ; 0334
   03       06   A8             01  E1 00020          BBC     #1, 6(FAB), 2$                            ; 0338
                         01A7   31 00025 1$:          BRW     35$                                       ;
   F8       22   AA             05  E0 00028 2$:      BBS     #5, 34(IFAB), 1$                          ;
                                53  D4 0002D          CLRL    R3                                        ; 0350
                 52      0200   8F  3C 0002F          MOVZWL  #512, R2                                  ;
                 51             01  D0 00034          MOVL    #1, R1                                    ;
                         0000G  30 00037              BSBW    RMSCACHE                                  ;
                 0B             50  E8 0003A          BLBS    STATUS, 4$                                ;
                           0C   AB  D5 0003D          TSTL    12(FAB)                                   ;
                                03  13 00040          BEQL    3$                                        ;
                         00F2   31 00042              BRW     20$                                       ;
                         00E6   31 00045 3$:          BRW     19$                                       ;
                         0000G  30 00048 4$:          BSBW    RMSCHKSUM                                 ; 0352
                 7C             50  E9 0004B          BLBC    STATUS, 11$                               ;
                 03        74   A5  B1 0004E          CMPW    116(BKT_ADDR), #3                         ; 0356
                                0C  1B 00052          BLEQU   5$                                        ;
                                53  D4 00054          CLRL    R3                                        ; 0359
                         0000G  30 00056              BSBW    RMSRELEASE                                ;
                 50      872C   8F  3C 00059          MOVZWL  #34604, R0                                ; 0360
                                6A  11 0005E          BRB     11$                                       ;
                                24  1E 00060 5$:      BGEQU   8$                                        ; 0366
                 50      00A0   CA  9E 00062          MOVAB   160(IFAB), R0                             ; 0368
   0F            60             01  E0 00067          BBS     #1, (R0), 6$                              ;
                 0C             60  E8 0006B          BLBS    (R0), 6$                                  ; 0370
   08            60             04  E0 0006E          BBS     #4, (R0), 6$                              ; 0372
   04            60             02  E0 00072          BBS     #2, (R0), 6$                              ; 0374
   0C            60             03  E1 00076          BBC     #3, (R0), 8$                              ; 0376
                                53  D4 0007A 6$:      CLRL    R3                                        ; 0379
                         0000G  30 0007C              BSBW    RMSRELEASE                                ;
                 50      8724   8F  3C 0007F          MOVZWL  #34596, R0                                ; 0380
                                65  11 00084 7$:      BRB     13$                                       ;
         00B7    CA        74   A5  90 00086 8$:      MOVB    116(BKT_ADDR), 183(IFAB)                  ; 0385
         00B0    CA        66   A5  B0 0008C          MOVW    102(BKT_ADDR), 176(IFAB)                  ; 0386
                           5C   AA  B4 00092          CLRW    92(IFAB)                                  ; 0388
         00B4    CA        14   A5  9B 00095          MOVZBW  20(BKT_ADDR), 180(IFAB)                   ; 0404
           0B    A5        0A   A5  91 0009B          CMPB    10(BKT_ADDR), 11(BKT_ADDR)                ; 0408
                                07  1B 000A0          BLEQU   9$                                        ;
           5E    AA        0A   A5  90 000A2          MOVB    10(BKT_ADDR), 94(IFAB)                    ; 0410
                                05  11 000A7          BRB     10$                                       ;
           5E    AA        0B   A5  90 000A9 9$:      MOVB    11(BKT_ADDR), 94(IFAB)                    ; 0412
                           10   AE  D4 000AE 10$:     CLRL    IDX_COMPR                                 ; 0416
                 7E             01  7D 000B1          MOVQ    #1, -(SP)                                 ; 0420
```

```
                                 55 DD 000B4          PUSHL    BKT_ADDR                                 :
                              0000G 30 000B6          BSBW     RM$AL_KEY_DESC                           :
                  5E             0C C0 000B9          ADDL2    #12, SP                                  :
                  56             50 D0 000BC          MOVL     R0, STATUS                               :
                  0A             56 E8 000BF          BLBS     STATUS, 12$                              :
                                 53 D4 000C2          CLRL     R3                                       :
                              0000G 30 000C4          BSBW     RM$RELEASE                               :
                  50             56 D0 000C7          MOVL     STATUS, R0                               :
                                 70 11 000CA 11$:     BRB      21$                                      :
            00B2  CA             01 90 000CC 12$:     MOVB     #1, 178(IFAB)                            : 0422
                  56             55 D0 000D1          MOVL     BKT_ADDR, KEY_DESC                       : 0424
                              0000V 30 000D4          BSBW     CHECK_TWO                                : 0430
              0C  AE             50 D0 000D7          MOVL     R0, STATUS                               :
                  0E       0C    AE E8 000DB          BLBS     STATUS, 14$                              :
                              0000G 30 000DF          BSBW     RM$CLOSE3                                :
                                 53 D4 000E2          CLRL     R3                                       :
                              0000G 30 000E4          BSBW     RM$RELEASE                               :
                  50       0C    AE D0 000E7          MOVL     STATUS, R0                               :
                                 4F 11 000EB 13$:     BRB      21$                                      :
        05        10  A6         03 E0 000ED 14$:     BBS      #3, 16(KEY_DESC), 15$                    : 0434
        04        10  A6         06 E1 000F2          BBC      #6, 16(KEY_DESC), 16$                    :
                  10  AE         01 D0 000F7 15$:     MOVL     #1, IDX_COMPR                            : 0436
                  04  AE   04    A6 9E 000FB 16$:     MOVAB    4(R6), 4(SP)                             : 0451
                                 66 D5 00100 17$:     TSTL     (KEY_DESC)                               : 0440
                                 03 12 00102          BNEQ     18$                                      :
                               00B7 31 00104          BRW      33$                                      :
              0C  AE             66 D0 00107 18$:     MOVL     (KEY_DESC), VBN                          : 0450
              08  AE   04        BE 3C 0010B          MOVZWL   @4(SP), OFFSET                           : 0451
                                 53 D4 00110          CLRL     R3                                       : 0455
                              0000G 30 00112          BSBW     RM$RELEASE                               :
                  29             50 E9 00115          BLBC     STATUS, 23$                              :
                                 53 D4 00118          CLRL     R3                                       : 0463
                  52  0200       8F 3C 0011A          MOVZWL   #512, R2                                 :
                  51    0C       AE D0 0011F          MOVL     VBN, R1                                  :
                              0000G 30 00123          BSBW.    RM$CACHE                                 :
                  15             50 E8 00126          BLBS     STATUS, 22$                              :
                         0C      A8 D5 00129          TSTL     12(FAB)                                  :
                                 09 12 0012C          BNEQ     20$                                      :
   0C    A8          50 00010000 8F C9 0012E 19$:     BISL3    #65536, STATUS, 12(FAB)                  :
                     50     C104 8F 3C 00137 20$:     MOVZWL   #49412, STATUS                           :
                                 62 11 0013C 21$:     BRB      30$                                      :
                              0000G 30 0013E 22$:     BSBW     RM$CHKSUM                                : 0466
                  5C             50 E9 00141 23$:     BLBC     STATUS, 30$                              :
        56        55       08    AE C1 00144 24$:     ADDL3    OFFSET, BKT_ADDR, KEY_DESC               : 0475
                              0000V 30 00149          BSBW     CHECK_TWO                                : 0481
                  6E             50 D0 0014C          MOVL     R0, STATUS                               :
                  05             6E E8 0014F          BLBS     STATUS, 25$                              :
                              0000G 30 00152          BSBW     RM$CLOSE3                                :
                                 41 11 00155          BRB      29$                                      :
                         00B2    CA 96 00157 25$:     INCB     178(IFAB)                                : 0485
                  50       14    A6 9A 0015B          MOVZBL   20(KEY_DESC), R0                         : 0489
            00B4  CA             50 B1 0015F          CMPW     R0, 180(IFAB)                            :
                                 06 1B 00164          BLEQU    26$                                      :
            00B4  CA       14    A6 9B 00166          MOVZBW   20(KEY_DESC), 180(IFAB)                  : 0491
              5E  AA       0A    A6 91 0016C 26$:     CMPB     10(KEY_DESC), 94(IFAB)                   : 0495
                                 05 1B 00171          BLEQU    27$                                      :
              5E  AA       0A    A6 90 00173          MOVB     10(KEY_DESC), 94(IFAB)                   : 0497
```

```
          5E    AA       0B    A6   91 00178 27$:     CMPB    11(KEY_DESC), 94(IFAB)           ; 0499
                                    05   1B 0017D           BLEQU   28$                              ;
          5E    AA       0B    A6   90 0017F           MOVB    11(KEY_DESC), 94(IFAB)           ; 0501
                         08    AE   DD 00184 28$:     PUSHL   OFFSET                           ; 0506
                         10    AE   DD 00187           PUSHL   VBN                              ;
                               56   DD 0018A           PUSHL   KEY_DESC                         ;
                            0000G   30 0018C           BSBW    RM$XL_KEY_DESC                   ;
                5E             0C   C0 0018F           ADDL2   #12, SP                          ;
                6E             50   D0 00192           MOVL    R0, STATUS                       ;
                0A             6E   E8 00195           BLBS    STATUS, 31$                      ;
                               53   D4 00198 29$:     CLRL    R3                               ;
                            0000G   30 0019A           BSBW    RM$RELEASE                       ;
                50             6E   D0 0019D           MOVL    STATUS, R0                       ;
                               30   11 001A0 30$:     BRB     36$                              ;
     04         10    A6       03   E1 001A2 31$:     BBC     #3, 16(KEY_DESC), 32$            ; 0510
                10    AE       01   D0 001A7           MOVL    #1, IDX_COMPR                    ; 0512
          04    AE       04    A6   9E 001AB 32$:     MOVAB   4(R6), 4(SP)                     ; 0516
          08    AE       04    BE   3C 001B0           MOVZWL  @4(SP), OFFSET                   ;
          0C    AE             66   D1 001B5           CMPL    (KEY_DESC), VBN                  ; 0522
                               89   13 001B9           BEQL    24$                              ;
                             FF42   31 001BB           BRW     17$                              ; 0470
                05       10    AE   E9 001BE 33$:     BLBC    IDX_COMPR, 34$                   ; 0529
          00B4  CA             02   A0 001C2           ADDW2   #2, 180(IFAB)                    ; 0531
                               53   D4 001C7 34$:     CLRL    R3                               ; 0535
                            0000G   30 001C9           BSBW    RM$RELEASE                       ;
                03             50   E9 001CC           BLBC    STATUS, 36$                      ;
                50             01   D0 001CF 35$:     MOVL    #1, R0                           ; 0537
                5E             14   C0 001D2 36$:     ADDL2   #20, SP                          ; 0539
                     00FC      8F   BA 001D5           POPR    #^M<R2,R3,R4,R5,R6,R7>           ;
                               05 001D9           RSB                                      ;

; Routine Size:  474 bytes,    Routine Base:  RM$RMS3 + 0000

;  478         0540  1
```

```
;  480     0541  1 %SBTTL  'CHECK_TWO'
;  481     0542  1 ROUTINE CHECK_TWO ( KEY_DESC : REF BBLOCK ) : RL$CHECK_TWO =
;  482     0543  1 !++
;  483     0544  1 !
;  484     0545  1 ! FUNCTIONAL DESCRIPTION:
;  485     0546  1 !
;  486     0547  1 !       Check to make sure that at least two records will fit in
;  487     0548  1 !       each index, if not don't even let the user open the file
;  488     0549  1 !       since it will only lead to trouble later
;  489     0550  1 !       note: create does check this but rms-11 doesn't
;  490     0551  1 !       if we release w/ a new rms-11 that does there would be no way of
;  491     0552  1 !       creating such files and we could take the check out
;  492     0553  1 !
;  493     0554  1 ! CALLING SEQUENCE:
;  494     0555  1 !
;  495     0556  1 !       CHECK_TWO( KEY_DESC )
;  496     0557  1 !
;  497     0558  1 ! INPUT PARAMETERS:
;  498     0559  1 !
;  499     0560  1 !       KEY_DESC -- pointer to the on-disk key descriptor
;  500     0561  1 !
;  501     0562  1 ! IMPLICIT INPUTS:
;  502     0563  1 !
;  503     0564  1 !       FAB -- so that in case of an error, the guilty key of reference
;  504     0565  1 !              can be passed back in the stv
;  505     0566  1 !
;  506     0567  1 ! OUTPUT PARAMETERS:
;  507     0568  1 !       none
;  508     0569  1 !
;  509     0570  1 ! IMPLICIT OUTPUTS:
;  510     0571  1 !       none
;  511     0572  1 !
;  512     0573  1 ! ROUTINE VALUE:
;  513     0574  1 !
;  514     0575  1 !       KSI if two keys will not fit in the index
;  515     0576  1 !       rmssuc if they will
;  516     0577  1 !
;  517     0578  1 ! SIDE EFFECTS:
;  518     0579  1 !       none
;  519     0580  1 !
;  520     0581  1 !--
;  521     0582  1
;  522     0583  2     BEGIN
;  523     0584  2
;  524     0585  2     EXTERNAL REGISTER
;  525     0586  2         R_IFAB_STR,
;  526     0587  2         R_FAB_STR;
;  527     0588  2
;  528     0589  2     ! Make sure at least 2 keys will fit in the index level
;  529     0590  2     !
;  530     0591  2     LOCAL
;  531     0592  2         KEYSZ,          ! Size of key
;  532     0593  2         BYTES;          ! Number of bytes available in bucket
;  533     0594  2
;  534     0595  2     BYTES = ( .KEY_DESC [ KEY$B_IDXBKTSZ ] * 512 ) - BKT$C_OVERHDSZ - 1;
;  535     0596  2     KEYSZ = .KEY_DESC [ KEY$B_KEYSZ ];
;  536     0597  2
```

```
;  537     0598  2     IF .IFAB [ IFB$B_PLG_VER ] LSSU PLG$C_VER_3
;  538     0599  2     THEN
;  539     0600  3         BEGIN
;  540     0601  3         IF 2 * ( .KEYSZ + 2 + IRC$C_IDXPTRBAS + IRC$C_IDXOVHDSZ) GTRU .BYTES
;  541     0602  3         THEN
;  542     0603  4             BEGIN
;  543     0604  4             FAB [ FAB$L_STV ] = .KEY_DESC [ KEY$B_KEYREF ];
;  544     0605  4             RETURN RMSERR(KSI);
;  545     0606  3             END;
;  546     0607  3         END
;  547     0608  2     ELSE
;  548     0609  3         BEGIN
;  549     0610  3
;  550     0611  3         BYTES = .BYTES - 3;
;  551     0612  3
;  552     0613  3         IF .KEYSZ LEQU KEY$C_MAX_INDEX
;  553     0614  3         THEN
;  554     0615  4             BEGIN  ! fixed index record
;  555     0616  4
;  556     0617  4             IF 2 * ( .KEYSZ + 4 ) GTRU .BYTES
;  557     0618  4             THEN
;  558     0619  5                 BEGIN
;  559     0620  5                 FAB [ FAB$L_STV ] = .KEY_DESC [ KEY$B_KEYREF ];
;  560     0621  5                 RETURN RMSERR(KSI);
;  561     0622  4                 END;
;  562     0623  4             END
;  563     0624  3         ELSE
;  564     0625  4             BEGIN  ! variable index records
;  565     0626  4
;  566     0627  4             IF 2 * ( .KEYSZ + 4 + 2 ) GTRU .BYTES
;  567     0628  4             THEN
;  568     0629  5                 BEGIN
;  569     0630  5                 FAB [ FAB$L_STV]  = .KEY_DESC [ KEY$B_KEYREF ];
;  570     0631  5                 RETURN RMSERR(KSI);
;  571     0632  4                 END;
;  572     0633  3             END;
;  573     0634  2         END;
;  574     0635  2
;  575     0636  3     RETURN RMSSUC()
;  576     0637  3
;  577     0638  1     END;


                                    0C  BB 00000 CHECK_TWO:
                                                 PUSHR   #^M<R2,R3>                          ; 0542
                        50      0A  A6  9A 00002         MOVZBL  10(KEY_DESC), R0            ; 0595
                50      50          09  78 00006         ASHL    #9, R0, R0                  ;
                        53      F1  A0  9E 0000A         MOVAB   -15(R0), BYTES              ;
                        50      14  A6  9A 0000E         MOVZBL  20(KEY_DESC), KEYSZ         ; 0596
                52      50          01  78 00012         ASHL    #1, KEYSZ, R2               ; 0601
                        03    00B7  CA  91 00016         CMPB    183(IFAB), #3               ; 0598
                                    09  1E 0001B         BGEQU   1$                          ;
                        51      0A  A2  9E 0001D         MOVAB   10(R2), R1                  ; 0601
                        53          51  D1 00021         CMPL    R1, BYTES                   ;
```

```
                              15  11 00024        BRB     4$                               ;
             53               03  C2 00026 1$:    SUBL2   #3, BYTES                        ; 0611
             06               50  D1 00029        CMPL    KEYSZ, #6                        ; 0613
                              06  1A 0002C        BGTRU   2$                               ;
             50        08     A2  9E 0002E        MOVAB   8(R2), R0                        ; 0617
                              04  11 00032        BRB     3$                               ;
             50        0C     A2  9E 00034 2$:    MOVAB   12(R2), R0                       ; 0627
             53               50  D1 00038 3$:    CMPL    R0, BYTES                        ;
                              0C  1B 0003B 4$:    BLEQU   5$                               ;
     0C      A8        15     A6  9A 0003D        MOVZBL  21(KEY_DESC), 12(FAB)            ; 0630
             50      8784     8F  3C 00042        MOVZWL  #34692, R0                       ; 0631
                              03  11 00047        BRB     6$                               ;
             50               01  D0 00049 5$:    MOVL    #1, R0                           ; 0636
                              0C  BA 0004C 6$:    POPR    #^M<R2,R3>                       ; 0638
                                  05 0004E        RSB                                      ;

; Routine Size:  79 bytes,    Routine Base:  RM$RMS3 + 01DA

;  578          0639  1
;  579          0640  1 END
;  580          0641  1
;  581          0642  0 ELUDOM

;                                PSECT SUMMARY
;
;      Name                   Bytes                      Attributes
;
;  RM$RMS3                        553  NOVEC,NOWRT, RD , EXE,NOSHR, GBL, REL, CON, PIC,ALIGN(2)

;                          Library Statistics
;
;                                        -------- Symbols --------        Pages      Processing
;      File                              Total    Loaded   Percent      Mapped     Time
;
;  _$255$DUA28:[RMS.OBJ]RMS.L32;1        3109       67         2          154        00:00.4

;                              COMMAND QUALIFIERS
;
;      BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3OPEN/OBJ=OBJ$:RM3OPEN MSRC$:RM3OPEN/UPDATE=(ENH$:RM3OPEN)

; Size:          553 code + 0 data bytes
; Run Time:          00:14.8
```

```
; Elapsed Time:     00:39.5
; Lines/CPU Min:     2607
; Lexemes/CPU-Min: 17788
; Memory Used:  193 pages
; Compilation Complete
```

RM3NEXTRE
LIS

RM3OPEN
LIS

RM3POSRFA
LIS

RM3PCKUNP
LIS

RM3POSKEY
LIS

RM3POSSEQ
LIS